IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	§	
	§	
MICHAEL KAGAN ET AL.	§	
	§	
Serial No.: 10/052,413	§	
	§	
Filed: January 23, 2002	§	Group Art Unit: 2142
	§	
For: QUEUE PAIR CONTEXT CACHE	§	Attorney Docket: 3091/23
	§	
Examiner: Kelvin Y. Lin	§	

TRANSMITTAL OF APPEAL BRIEF

Commissioner of Patents and Trademarks
Washington, DC 20231

Dear Sir:

Transmitted herewith in triplicate is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed on April 13, 2006.

The application is on behalf of

___ other than a small entity

_X_ small entity

verified statement:

___ attached

_X_ already filed

Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is:

_X_	small entity	$ 250
___	other than a small entity	$ 500

Appeal Brief fee due $250

___ Applicant petitions for an extension of time under 37 CFR 1.136 for the total number of months checked below:

	small entity	not small entity
___one month	$ 55	$ 110
___two months	$ 215	$ 430
___three months	$ 490	$ 980
___four months	$ 765	$ 1530

If an additional extension of time is required please consider this a petition therefor.

The total fee due is:

Appeal brief	$____
Extension fee (if any)	$____
TOTAL FEE DUE	$____

Please charge Account No. 06-2140 the sum of $____. A duplicate copy of this transmittal letter is attached.

If any additional extension and/or fee is required, this is a request therefor and to charge Account No. 06-2140.

If any additional fee for claims is required, please charge Account No. 06-2140.

Respectfully submitted,

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

ZZW
AF

OIPE
JUN 12 2006
IAP38
PATENT & TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	§	
MICHAEL KAGAN ET AL.	§	
Serial No.: 10/052,413	§	
Filed: January 23, 2002	§	Group Art Unit: 2142
For: QUEUE PAIR CONTEXT CACHE	§	Attorney Docket: 3091/23
Examiner: Kelvin Y. Lin	§	

Commissioner of Patents and Trademarks
Washington, DC 20231
ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF

Dear Sir:

This is in furtherance of the Notice of Appeal filed in this case on April 13, 2006.

The fees required under § 1.17(f) and any required petition for extension of time for filing this brief and fees therefor are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate.

This brief contains these items under the following headings and in the order set forth below:

I. REAL PARTY IN INTEREST

II. RELATED APPEALS AND INTERFERENCES

III. STATUS OF CLAIMS

IV. STATUS OF AMENDMENTS

I. REAL PARTY IN INTEREST

The real party in interest in this case is:

Mellanox Technologies Ltd.

P. O. Box 586

20692 Yokneam

ISRAEL

II. RELATED APPEALS AND INTERFERENCES

NONE

III. STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 49

B. STATUS OF ALL THE CLAIMS

1. Claims cancelled: NONE
2. Claims withdrawn from consideration but not cancelled: 38-49
3. Claims pending: 1-37
4. Claims allowed: NONE
5. Claims rejected: 1-37

C. CLAIMS ON APPEAL

The claims on appeal are: 1-37

IV. STATUS OF AMENDMENTS

No claims were amended subsequent to the final rejection.

V. SUMMARY OF INVENTION

Independent claim 1 recites a network interface adapter (Figures 1 and 2, HCA 22) that includes a network interface, a host interface, packet processing circuitry and a cache memory.

The network interface (Figure 2, IB output port 56 and IB input port 58) sends and receives data packets over a network (Figure 1, IB fabric 26).

The host interface (Figure 2 and page 20 lines 13-19: TPT 64) couples the network interface adapter to a host processor (Figure 1, CPU 24) and to a system memory (Figure 1, system memory 38). The system memory contains context information (Figure 1, QP context 42) for transport service instances (page 2 lines 4-5: "queue pairs) used to send and receive the data packets over the network. Each data packet belongs to a respective service instance.

The packet processing circuitry (Figure 2: doorbells 50, execution unit 52, send data engine 54, transport check unit 60, receive data engine 62) is coupled between the network interface and the host interface. The packet processing circuitry processes the data packets using the context information of the respective service instances (page 20 line 31 through page 21 line 14).

The cache memory (Figure 2, cache memory 66) loads from the system memory, and stores, context information for the transport service instances of the data packets being processed by the packet processing circuitry (page 21 lines 6-14).

Dependent claim 2 adds to claim 1 the limitations that the transport service instances have respective instance numbers (page 22 lines 28-30) and that the cache memory includes tables (Figure 3, tables 80-84) with entries in which the context information of the transport service instances is indexed by a portion of the service instance numbers (page 22 lines 30-31).

Dependent claim 10 adds to claim 1 the limitation that the context information in the cache memory includes fields that are updated by the packet processing circuitry in the course of processing the data and that are copied back to the context information in the system memory after the packets have been processed (page 24 lines 3-6).

Independent claim 17 recites a method of network communications.

A network adapter (Figures 1 and 2, HCA 22) having a cache memory (Figure 2, cache memory 66) is coupled between a host processor (Figure 1, CPU 24) and a network (Figure 1, IB fabric 26).

Context information (Figure 1, QP context 42) is stored in a system memory (Figure 1, system memory 38) associated with the host processor. The context information relates to transport service instances (page 2 lines 4-5: "queue pairs") used in sending and receiving data packets over the network via the adapter. Each data packet belongs to a respective service instance.

Responsive to the use of some of the transport service instances to send and receive data packets, context information relating to those transport service instances is loaded into the cache memory (page 29 lines 5-17, Figure 4A blocks 104 and 106). The data packets are processed using the adapter based on the context information in the cache memory (page 29 lines 18-23, Figure 4A block 108).

Dependent claim 18 adds to claim 17 the limitations that the transport service instances have respective instance numbers (page 22 lines 28-30) and that loading the context information into the cache memory includes using entries in tables (Figure 3, tables 80-84) to provide access to the context information. The entries are indexed by a portion of the service instance numbers (page 22 lines 30-31).

Independent claim 33 recites a host channel adapter (Figures 1 and 2, HCA 22) that includes a fabric interface, a host interface, packet processing circuitry and a cache memory.

The fabric interface (Figure 2, IB output port 56 and IB input port 58) is coupled to send and receive data packets over a switch fabric (Figure 1, IB fabric 26).

The host interface (Figure 2 and page 20 lines 13-19: TPT 64) is for coupling to a host processor (Figure 1, CPU 24) and to a system memory (Figure 1, system memory 38) associated with the host processor. The system memory contains context information (Figure 1, QP context 42) with respect to queue pairs (page 2 lines 4-5). Each data packet belongs to a respective queue pair.

The packet processing circuitry (Figure 2: doorbells 50, execution unit 52, send data engine 54, transport check unit 60, receive data engine 62) is coupled between the fabric interface and the host interface. The packet processing circuitry processes the data packets using the context information of the respective queue pairs (page 20 line 31 through page 21 line 14).

The cache memory (Figure 2, cache memory 66) loads from the system memory, and stores, context information for the queue pairs of the data packets being processed by the packet processing circuitry (page 21 lines 6-14).

Independent claim 37 recites a method of network communications.

A host channel adapter (Figures 1 and 2, HCA 22) having a cache memory (Figure 2, cache memory 66) is coupled between a host processor (Figure 1, CPU 24) and a switch fabric (Figure 1, IB fabric 26).

Context information (Figure 1, QP context 42) is stored in a system memory (Figure 1, system memory 38) associated with the host processor. The context information relates to queue pairs (page 2 lines 4-5) used in sending and receiving

data packets over the fabric. Each data packet belongs to a respective service instance.

Responsive to the use of some of the queue pairs to send and receive data packets, context information relating to those queue pairs is loaded into the cache memory (page 29 lines 5-17, Figure 4A blocks 104 and 106). The data packets are processed using the adapter based on the context information in the cache memory (page 29 lines 18-23, Figure 4A block 108).

VI. ISSUES

Whether claims 1, 17, 33 and 37 are unpatentable under 35 USC 102(e) over Starr et al., US Patent No. 6,807,581 (henceforth, "Starr et al. '581).

Whether claims 2-16, 18-32 and 34-36 are unpatentable under 35 USC 103(a) over Starr et al. '581 in view of Dobbins et al., US Patent No. 5,509,123 (henceforth, "Dobbins et al. '123")

VII. ARGUMENTS

VIIIA ARGUMENTS - REJECTIONS UNDER 35 USC 102

Starr et al. '581 teach an intelligent network interface card (INIC) (e.g., INIC **22** of Figure 1) that relieves the CPU of a host (e.g. CPU **30** of host **20** of Figure 1) of the burden of network communication protocol processing. For each message that is to be exchanged between the host and a remote host, a communication control block (CCB) is created. The CCBs are stored in a CCB cache in the INIC (e.g. in CCB cache **74** of INIC **22** of Figure 1).

Comparing the teachings of Starr et al. '581, as illustrated in Figure 1 of that patent, reveals the following correspondences to the elements of the present invention:

PHY **58** and MAC **60** correspond to the network interface of the present invention.

I/O bridge **50** corresponds to the host interface of the present invention.

CPU **30** corresponds to the host processor of the present invention.

Host storage unit **66** corresponds to the system memory of the present invention.

Processor **44** corresponds to the packet processing circuitry of the present invention.

CCB cache **74** corresponds to the cache memory of the present invention, because the CCBs correspond to the context information of the present invention. That the CCBs of Starr et al. '581 correspond to the context information of the present invention is abundantly clear from a comparison of the examples of context information given in the specification (page 3 lines 17-20: destination address, negotiated operating limits, service level, keys for access control; page 3 line 21: current packet sequence number) to the examples of the contents of a CCB that are

given by Starr et al. '581 (column 7 lines 30-31: source and destination addresses and ports; column 7 lines 32-36: source and destination MAC addresses, source and destination IP addresses, source and destination TCP ports, TCP variables such as timers and receive and transmit windows for sliding window protocols; column 7 lines 60-63: state information regarding the message, such as the length of the message and the number and order of packets that have been processed).

One crucial difference between the teachings of Starr et al. '581 and the present invention is that the present invention, as recited in claims 1, 17, 33 and 37, stores the context information *both* in the system memory *and* in the cache memory. Indeed, the problem solved by the present invention is that, with "16 million QPs allowed by the IB specification" (page 4 lines 15-16), most of the context information of the present invention *must* be stored in the system memory. By contrast, Starr et al. '581 store the CCBs *only* in the CCB cache and *not* in the host storage unit.

Thus, the present invention, as recited in claims 1, 17, 33 and 37, is not anticipated by Starr et al. '581. Furthermore, the present invention, as recited in claims 1, 17, 33 and 37, is not even obvious from Starr et al. '581. There is neither a hint nor a suggestion in Starr et al. '581 of any need to store CCBs in the host storage unit. Hence, claims 1, 17, 33 and 37 are allowable in their present form over the prior art cited by the Examiner.

Applicant presented these arguments in response to the Office Action mailed June 9, 2005. In the Office Action mailed December 13, 2005, the Examiner responded to these arguments by citing Starr et al. '581 column 8 lines 6-15 as teaching the storing of certain types of data in both system memory (host file cache **24**, host storage unit **66**) and cache memory (INIC file cache **80**, INIC storage unit **70**). But a close reading of Starr et al. '581 column 8 lines 7-17:

> Upon matching the packet summary with the CCB, assuming no exception conditions exist, <u>the data of the packet</u>, without network or transport layer headers, is sent by direct memory access (DMA) unit **68** to the destination in file cache **80** or file cache **24** denoted by the CCB.
>
> At some point after all <u>the data from the message</u> has been cached as a file stream in INIC file cache **80** or host file cache **24**, <u>the file stream of data</u> is then sent, by DMA unit **68** under control of the file system **23**, from that file cache to the INIC storage unit **70** or host storage unit **66**, under control of the file system. (emphasis added)

shows that the data that are stored in both system memory and cache memory, are not context information about the connections along which the packets are exchanged (the CCBs) but rather data of the packets themselves. Therefore, the cited portion of Starr et al. '581 has nothing whatsoever to do with the present invention.

In an Advisory Action mailed March 27, 2006, The Examiner responded as follows:

> In col. 17, l.1-15, Starr discloses the protocol stack in charge of the connection creation and release hands out/ receives CCB for fast-path connection to/from the INIC. Furthermore, in Fig. 14, and col. 17, l.36-57, Starr discloses the server and client have fast-path with Gigabit Ethernet compliant for communication processing includes the data link layer, IP layer, and TCP layer. A server attached storage unit 640 connect to INIC over the network line and has a slow-path and fast-path communication processing that travel over the INIC 622 and a file cache. Therefore, Starr does disclose the context of connection information have been stored in cache and network storage unit.

Applicant respectfully disagrees. Column 17 lines 1-17 of Starr et al. '581 are as follows:

> The protocol stack **38** in this embodiment includes data link layer **562**, network layer **564**, transport layer **566**, upper layer interface **568** and upper layer **570**. The upper layer **570** may represent a session, presentation and/or application layer, depending upon the particular protocol employed and message communicated. The protocol stack **38** processes packet headers in the slow-path, creates and tears down connections, hands out CCBs for fast-path connections to the INIC, and <u>receives CCBs for fast-path connections being flushed from the INIC to the host **20**.</u> The upper layer interface **568** is generally responsible for assembling CCBs based upon connection and status information created by the data link layer **562**, network layer **564** and transport layer **566**, and handing out the CCBs to the INIC via the

> INIC device driver **560**, or <u>receiving flushed CCBs from the INIC via the INIC device driver **560**</u>. (emphasis added)

So the issue of whether CCBs are stored outside the INIC turns on whether the "flushed" CCBs are stored in host **20**.

The reason for "flushing" CCBs to host **20** is described in column 32 line 60 through column 33 line 3:

> If the software executed by processor **780** detects one of these exception conditions, then processor **780** determines that the "fast-path candidate" is not a "fast-path packet." In such a case, the connection context for the packet is "flushed" (the connection context is passed back to the host) so that the connection context is no longer present in the list of connection contexts under control of INIC **22**. The entire packet (headers and data) is transferred to a buffer in host **20** for "slow-path" transport layer and network layer processing by the protocol stack of host **20**.

Column 7 lines 19-22 gives the reason for "slow-path" processing:

> Slow-path processing allows any packets that are not conveniently transferred by the fast-path of the INIC **22** to be processed <u>conventionally</u> by the host **20**. (emphasis added)

An example of "conventional" processing is given in column 2 lines 21-62:

> An example of conventional processing of a network message such as a file transfer illustrates some of the processing steps that slow network data storage. A network interface card (NIC) typically provides a physical connection between a host and a network or networks, as well as providing media access control (MAC) functions that allow the host to access the network or networks. When a network message packet sent to the host arrives at the NIC, MAC layer headers for that packet are processed and the packet undergoes cyclical redundancy checking (CRC) in the NIC. The packet is then sent across an input/output (I/O) bus such as a peripheral component interconnect (PCI) bus to the host, and stored in host memory. The CPU then processes each of the header layers of the packet sequentially by running instructions from the protocol stack. This requires a trip across the host memory bus initially for storing the packet and then subsequent trips across the host memory bus for sequentially processing each header layer. After all the header layers for that packet have been processed, the payload data from the packet is grouped in a file cache with other similarly-processed payload packets of the message. The data is reassembled by the CPU according to the file system as file blocks for storage on a disk or disks. After all the packets have been processed and the message has been reassembled as

> file blocks in the file cache, the file is sent, in blocks of data that may be each built from a few payload packets, back over the host memory bus and the I/O bus to host storage for long term storage on a disk, typically via a SCSI bus that is bridged to the I/O bus.
>
> Alternatively, for storing the file on a SAN, the reassembled file in the file cache is sent in blocks back over the host memory bus and the I/O bus to an I/O controller configured for the SAN. For the situation in which the SAN is a FC network, a specialized FC controller is provided which can send the file blocks to a storage device on the SAN according to Fibre Channel Protocol (FCP). For the situation in which the file is to be stored on a NAS device, the file may be directed or redirected to the NAS device, which processes the packets much as described above but employs the CPU, protocol stack and file system of the NAS device, and stores blocks of the file on a storage unit of the NAS device.

This description of "conventional" processing is silent concerning the contexts of the processed packets. As best understood, in "conventional" processing, each packet's context is determined from the packet itself and is not saved for subsequent packets. It now is clear why the CCB of a "slow-path" packet is "flushed" to host **20** and what happens to the CCB at host **20**. The CCB is flushed to host **20** in order to spare host **20** the effort of determining whatever about the packet's context host **20** needs to know for conventional processing because the context already was determined when the CCB was created and stored in CCB cache **74**. After the packet is processed conventionally, the CCB is discarded, not saved.

In short, the *only* place where Starr et al. '581 save CCBs is CCB cache **74**.

VIIIB ARGUMENTS - REJECTIONS UNDER 35 USC 103

It is demonstrated above, in the context of the rejections under 35 USC 102, that independent claims 1, 17 and 33 are allowable in their present form. It follows that claims 2-16, 18-31 and 34-36, that depend therefrom, also are allowable.

Although claims 2-16, 18-32 and 34-36 are allowable merely by virtue of depending from claims 1, 17 and 33, there are additional reasons why some of these claims are allowable over the prior art cited by the Examiner. Specifically, the Examiner argued that the features of the present invention that are recited in claims 2, 10 and 18 are taught by Dobbins et al. '123. The following rebuttal of the Examiner's argument was presented in the responses to the Office Actions mailed August 26, 2004 and June 9, 2005. The Examiner did not address these arguments in the Advisory Action mailed March 27, 2006..

Dobbins et al. '123 teach an object-oriented architecture for a network router. According to the OSI model, a network router works in layer 3, the "network layer". Architecture **200** of routers **105**, **106**, **107** and **108** includes a forwarding engine **203** that in turn includes forwarding engine objects such as host forwarding engine object **232** and protocol forwarding engine object **234**. Because deciding how to forward a packet along the network is computationally intensive, forwarding engine **203** uses a cache to save forwarding strategies for subsequent packets that share the same source and destination addresses as the packets for which the forwarding strategies were computed. The forwarding strategies are indexed according to destination addresses (column 7 line 62).

How Dobbins et al. '123 index their cache is described in column 7 line 62. The index of the information in the cache for a data packet is the destination address

of the data packet. By contrast, the indices of the present invention, as recited in claims 2 and 18, are portions of the service instance numbers.

Turning now to the direction of data flow with respect to the cache, in Dobbins et al. '123, this data flow is only into the cache. The Examiner cited Dobbins et al. '123 column 7 line 67 as anticipating claim 10, which recites fields in the cache memory that are '"updated by the packet processing circuitry" and then "copied back to the context information in the system memory", *i.e.*, data flow from the cache back to the host. The full citation from Dobbins et al. '123 is column 7 lines 64-67:

> ...if the destination network address is not located in cache memory, accessing a forward look-up table **20** for the best route to the destination network address, and then <u>updating its cache</u>. (emphasis added)

The cache in question clearly is the cache of the forwarding engine, so that the data flow is from the host to the cache.

Respectfully submitted,

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

Date: May 31, 2006

VIII. APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

The text of the claims on appeal is:

1. A network interface adapter, comprising:

a network interface, coupled to send and receive data packets over a network;

a host interface, for coupling to a host processor and to a system memory associated therewith, the system memory containing context information with respect to a plurality of transport service instances used to send and receive the data packets over the network, each of the data packets belonging to a respective one of the service instances;

packet processing circuitry, coupled between the network interface and the host interface, and adapted to process the data packets using the context information of the respective service instances; and

a cache memory associated with the packet processing circuitry and coupled to load from the system memory and store the context information of the respective transport service instances for the data packets being processed by the packet processing circuitry.

2. An adapter according to claim 1, wherein the transport services instances have respective instance numbers, and wherein the cache memory comprises one or more tables having entries indicating the context information of the respective transport service instances, indexed by a portion of the service instance numbers.

3. An adapter according to claim 2, wherein the portion of the instance numbers comprises a predetermined number of the least significant bits of the instance numbers.

4. An adapter according to claim 2, wherein the one or more tables comprise at least two tables.

5. An adapter according to claim 2, wherein the entries comprise respective target fields, corresponding to at least a segment of the service instance numbers of the service instances to which the entries belong, and wherein the target fields are compared to the segment of the service instance numbers of the data packets to determine that a cache hit has occurred, whereupon the packet processing circuitry reads the context information from one of the tables.

6. An adapter according to claim 5, wherein when the cache hit does not occur, the context information is read from the system memory and loaded into the cache memory.

7. An adapter according to claim 1, wherein the packet processing circuitry comprises a cache controller, which is adapted, responsive to a request from the circuitry to access the context information in the cache memory with respect to one of the service instances, to determine whether a cache hit has occurred, and when the cache hit has not occurred, to read the requested context information from the system memory and load the requested context information into the cache memory in place of the context information of another one of the service instances.

8. An adapter according to claim 7, wherein the context information is organized in the cache memory using a plurality of tables having entries referenced by respective indices, and wherein the cache controller is adapted, while reading the requested context information from the system memory for one of the service instances having a given one of the indices, to block access by the packet processing circuitry to the context information of the service instances having the given one of the indices, while enabling the packet processing circuitry to access the context information of the service instances with other indices.

9. An adapter according to claim 7, wherein the cache controller is adapted, responsive to the request to access the context information, to set a flag with respect to the service instance for which the context information is requested indicating that the context information is in use, and wherein the cache controller is further adapted, upon loading the context information into the cache memory, to store the loaded context information in the cache in place of the context information of another one of the service instances whose flag is not set.

10. An adapter according to claim 1, wherein the context information loaded into the cache memory comprises one or more fields that are updated by the packet processing circuitry in the course of processing the data packets, and wherein the updated fields are copied back to the context information in the system memory after the data packets have been processed.

11. An adapter according to claim 10, wherein the updated fields comprise packet serial numbers of packets processed by the circuitry.

12. An adapter according to claim 1, wherein the context information stored in the cache memory comprises a send cache, containing the context information pertaining to packets generated responsive to requests submitted by the host processor, and a receive cache, containing the context information pertaining to packets generated responsive to requests submitted to the adapter by remote entities over the network.

13. An adapter according to claim 1, wherein the packet processing circuitry comprises:

an outgoing packet generator, adapted to generate the packets for delivery to remote entities via the network; and

an incoming packet processor, coupled to receive and process the packets from the remote entities via the network,

wherein both the outgoing packet generator and the incoming packet processor are coupled to access the same context information in the cache memory.

14. An adapter according to claim 13, wherein the outgoing packet generator is adapted to generate the packets for delivery to the remote entities responsive both to outgoing requests submitted by the host processor via the host interface and to incoming requests conveyed by the packets received from the remote entities.

15. An adapter according to claim 14, wherein the incoming packet processor is adapted to process both the packets that are received from the remote entities responsive to the outgoing requests conveyed by the packets delivered to the

remote entities and the packets that are received from the remote entities conveying the incoming requests.

16. An adapter according to claim 1, wherein the transport service instances comprises queue pairs, which are used to interact with a transport layer of the network.

17. A method for network communications, comprising:

coupling a network adapter having a cache memory between a host processor and a network;

storing context information in a system memory associated with the host processor, the context information relating to a plurality of transport service instances for use in sending and receiving data packets over the network via the adapter, each of the data packets belonging to a respective one of the service instances;

responsive to the use of a subset of the transport service instances to send and receive the data packets, loading into the cache memory the context information relating to the transport service instances in the subset; and

processing the data packets using the adapter based on the context information in the cache memory.

18. A method according to claim 17, wherein the transport services instances have respective instance numbers, and wherein loading the context information into the cache memory comprises using respective entries in one or more tables in the cache memory to provide access to the context information, wherein the entries are indexed by a portion of the service instance numbers.

19. A method according to claim 18, wherein the portion of the instance numbers comprises a predetermined number of the least significant bits of the instance numbers.

20. A method according to claim 18, wherein storing the context information in the one or more tables comprises storing the context information in at least two tables.

21. A method according to claim 18, wherein storing the context information comprises storing at least a segment of the respective service instance numbers of the service instances to which the entries belong in respective target fields of the entries, and wherein processing the data packets comprises comparing the target fields to the segment of the service instance numbers of the data packets to determine that a cache hit has occurred, whereupon the packet processing circuitry reads the context information from one of the tables.

22. A method according to claim 21, wherein loading the context information comprises, when the cache hit does not occur, reading the context information from the system memory and loading the information read from the system memory into the cache memory.

23. A method according to claim 17, wherein loading the context information into the cache memory comprises receiving a request to access the context information in the cache memory with respect to one of the service instances, and upon determining that a cache hit has not occurred, reading the requested context

information from the system memory, and storing the requested context information into the cache memory in place of the context information of another one of the service instances.

24. A method according to claim 23, wherein loading the context information into the cache memory comprises organizing the context information in the cache memory using a plurality of tables having entries referenced by respective indices, and comprising, while reading the context information from the system memory for one of the service instances having a given one of the indices, blocking access to the context information of the service instances having the given one of the indices, while allowing access the context information of the service instances with other indices.

25. A method according to claim 23, wherein loading the context information read from the system memory into the cache memory comprises setting a flag with respect to the service instance for which the context information is requested indicating that the context information is in use, and wherein storing the requested context information comprises storing the requested context information in place of the context information of another one of the service instances whose flag is not set.

26. A method according to claim 17, wherein processing the data packets comprises updating one or more fields of the context information in the cache memory, and comprising copying the updated fields back to the context information in the system memory after the data packets have been processed.

27. A method according to claim 26, wherein updating the one or more fields comprises writing to the cache memory packet serial numbers of packets processed by the circuitry.

28. A method according to claim 17, wherein loading the context information comprises loading into a send cache the context information pertaining to packets generated responsive to requests submitted by the host processor, and loading into a receive cache the context information pertaining to packets generated responsive to requests submitted to the adapter by remote entities over the network.

29. A method according to claim 17, wherein processing the data packets comprises both generating some of the packets for delivery to remote entities via the network and processing others of the packets received from the remote entities via the network, using the same context information in the cache memory.

30. A method according to claim 29, wherein generating the packets comprises generating the packets responsive both to outgoing requests submitted by the host processor and to incoming requests conveyed by the packets received from the remote entities, using the context information in the cache memory.

31. A method according to claim 30, wherein processing the packets that are received from the remote entities comprises receiving and processing both the packets returned by the remote entities responsive to the outgoing requests in the packets delivered to the remote entities and the packets that are received from the

remote entities conveying the incoming requests, using the context information in the cache memory.

32. A method according to claim 17, wherein the transport service instances comprises queue pairs, which are used to interact with a transport layer of the network.

33. A host channel adapter, comprising:

a fabric interface, coupled to send and receive data packets over a switch fabric;

a host interface, for coupling to a host processor and to a system memory associated therewith, the system memory containing context information with respect to a plurality of queue pairs, each of the data packets belonging to a respective one of the queue pairs;

packet processing circuitry, coupled between the fabric interface and the host interface, and adapted to process the data packets using the context information of the respective queue pairs; and

a cache memory associated with the packet processing circuitry and coupled to load from the system memory and store the context information of the respective queue pairs for the data packets being processed by the packet processing circuitry.

34. An adapter according to claim 33, wherein the packet processing circuitry comprises:

an execution unit, adapted to generate the packets for delivery to remote entities via the network; and

a transport check unit, coupled to process the packets received from the remote entities via the network,

wherein both the execution unit and the transport check unit are coupled to access the same context information in the cache memory.

35. An adapter according to claim 34, wherein the execution unit is adapted to generate the packets for delivery to the remote entities responsive both to work queue entries submitted by the host processor via the host interface and to incoming requests conveyed by the transport check unit to the execution unit, in response to the packets received from the remote entities.

36. An adapter according to claim 35, wherein the transport check unit is adapted to process both the packets that are received from the remote entities in response to the packets sent over the network responsive to the work queue items and the packets that are received from the remote entities conveying the incoming requests.

37. A method for network communications, comprising:

coupling a host channel adapter having a cache memory between a host processor and a switch fabric;

storing context information in a system memory associated with the host processor, the context information relating to a plurality of queue pairs for use in sending and receiving data packets over the fabric, each of the data packets belonging to a respective one of the queue pairs;

responsive to the use of a subset of the queue pairs to send and receive the data packets, loading into the cache memory the context information relating to the queue pairs in the subset; and

processing the data packets using the adapter based on the context information in the cache memory.

IX. APPENDIX OF EVIDENCE

NONE

X. APPENDIX OF RELATED PROCEEDINGS

NONE